Sponsor Award #NA02-1472

Final Report
Effective Team Support: From Modeling to Software Agents

Submitted by

Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213

Principal Investigator:	Bonnie John	(412) 268-4469	bej@cs.cmu.edu
Co-Investigator:	Katia Sycara	(412) 268-8825	katia@cs.cmu.edu

1. Abstract

The purpose of this research contract was to perform multidisciplinary research between CMU psychologists, computer scientists and engineers and NASA researchers to design a next generation collaborative system to support a team of human experts and intelligent agents. To achieve robust performance enhancement of such a system, we had proposed to perform task and cognitive modeling to thoroughly understand the impact technology makes on the organization and on key individual personnel. Guided by cognitively-inspired requirements, we would then develop software agents that support the human team in decision making, information filtering, information distribution and integration to enhance team situational awareness. During the period covered by this final report, we made substantial progress in modeling infrastructure and task infrastructure. Work is continuing under a different contract to complete empirical data collection, cognitive modeling, and the building of software agents to support the teams' task.

1 Cognitive Modeling Infrastructure

Cognitive modeling is a potentially powerful tool for predicting human interaction with computer systems (Pew and Mavor, 1998), but it has historically been a difficult and time-consuming task, requiring researchers with doctorates in psychology and/or computer science. As part of this research contract, we made substantial progress towards making cognitive modeling more accessible and tractable for modeling the knowledge required to operate complex systems.

Our first effort was to implement a well-established modeling framework, called CPM-GOMS (John, 1990), in a cognitive architecture developed at NASA, called Apex (Freed, 1998). CPM-GOMS is a modeling method that combines the task decomposition of a GOMS analysis with a model of human resource usage at the level of cognitive, perceptual, and motor operations. CPM-GOMS models have made accurate predictions about skilled user behavior in routine tasks, but developing such models is tedious and error-prone. In the process of implementing CPM-GOMS in Apex, we discovered ways to automate some of the harder aspects of CPM-GOMS modeling. This breakthrough is documented in a CHI 2002 paper (John, Vera, Matessa, Freed, & Remington, 2002, reproduced in Appendix A) where we describe the process for automatically generating CPM-GOMS models from a hierarchical task decomposition expressed in Apex. Resource scheduling in Apex automates the difficult task of interleaving the cognitive, perceptual, and motor resources underlying common task operators (e.g. mouse move-and-click). Apex's UI automatically generates PERT charts, which allow modelers to visualize a model's complex parallel behavior. Because interleaving and visualization is now automated, it is feasible to construct arbitrarily long sequences of behavior.

We further refined the theoretical ideas underlying our success in automating CPM-GOMS in a paper at the Cognitive Science Conference (Matessa, Vera, John, Remington & Freed, 2002, reproduced in Appendix B). In it, we explored the concept of reusable templates of common behaviors and their efficacy for generating zero-

parameter *a priori* predictions of complex human behavior. This paper detailed the features we believe are important when moving from hand-crafted models of particular tasks to reusable building blocks of commonly occurring behavior. As this becomes common practice, proportionately more attention can be paid to the task analysis specific to each new domain. We also taught a tutorial at the Cognitive Science Conference about how to use Apex for different types of predictive cognitive modeling: KLM, GOMS and CPM-GOMS (Remington, John, Vera, Matessa, Freed, Dalal, Harris & Dahlman, 2002).

In addition to the theoretical progress, we also contributed to modeling infrastructure by supervising Carnegie Mellon University Masters of Human-Computer Interaction and Masters of Software Engineering project classes. The charge to the students was to "make a cognitive modeler's life easier". The students built prototypes of tools to visualize what complex cognitive models are doing, and also to build working tools for constructing simple interfaces with which the cognitive models interacted. The latter tools were sufficiently robust to be included in the CogSci2002 tutorial lecture and running code was distributed to the tutorial attendees.

2 Task Infrastructure

MORSE is being developed for NASA under a this contract, as a platform to explore team performance, software agents and cognitive modeling. MORSE is an environment for three team members, encompassing some of the challenges facing Range Operations at KSC. It is a distributed system that simulates a task performed by a team of three human operators, each being responsible for some aspect of ensuring that launching a space vehicle from KSC will not endanger the general populace. MORSE provides monitoring stations at Cape Canaveral (Figure 1), Antigua and the Ascension Islands, where human operators track the progress of the mission in the 15 hours (scaled to be 15 minutes of real time in the simulation) prior to launch, and decide whether to abort or go ahead with the launch. To make this decision, the team must monitor weather and incursions (e.g., aircraft) that have entered an area bounded by launch impact lines. The overall team objective is to effect a safe launch, where no severe weather threatens the launch vehicle and no incursions are left within the impact lines at launch time. The human team has at its disposal weather balloons that can get current weather data, radars that allow the team members to see incursions in the areas covered by the radars, and interceptor vehicles that can be appointed to intercept incursions. Weather balloons, radars, and interceptor vehicles are shared resources that team members must acquire through coordination with each other and utilize for the performance of the team task, i.e. the safe launch, or abortion of a launch that is predicted to be unsafe.

The MORSE task has several important features that make it suitable for the proposed research. It has many features in common with actual Range Operations, including uncertain data, predictions based on old data, a potentially heavy workload (e.g., many incursions), time pressure, and the necessity for communication between team members. Because it is based on a real NASA task, it includes individual performance issues important to NASA. These include judging the trajectory of a

moving object on a radar screen, extrapolating and interpolating from incomplete or uncertain data (to judge the weather), and communicating about a shared space. In addition, it incorporates important aspects of teamwork including varying task complexity, limited resources, and communication between team members in order to allocate resources and arrive at a team decision (to go ahead with the launch or not).

In addition to the human performance aspects of the task, MORSE also has important technical features. MORSE was designed to be a very flexible platform for conducting team experiments, giving the experimenter extensive control over the scenarios seen by the team. Because it is fully instrumented, MORSE allows some direct measurement of collaboration success and failure (e.g., two team members attempting to escort the same incursion out of the impact area), as well as more traditional measures of individual and team performance. MORSE's architecture makes it easy to incorporate software agents into the task. It also communicates with cognitive modeling architectures making it easy to do GOMS modeling in Apex [6] or learning models in ACT-R [1]. It presents the same information and control opportunities to cognitive models as it does to human participants, logs behavior from both as well as any software agents, and allows playback, making analysis of data as easy as possible.

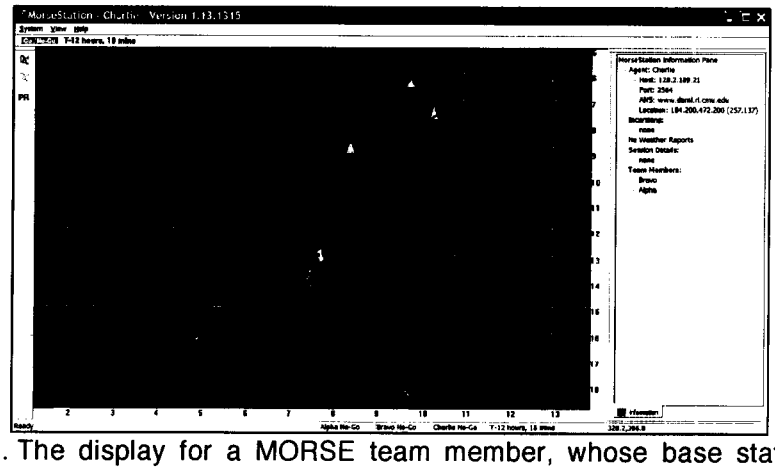

Figure 1. The display for a MORSE team member, whose base station is located at Cape Canaveral. Each team member sees a similar display, showing a different base station and view of the map.

3 Ongoing work

The work described in this final report is on-going at CMU and NASA Ames Research Center. Data collection of teams performing the MORSE task will begin in the Fall of 2003, and cognitive modeling and software agent construction will follow. Evaluation of the effectiveness of software agents to support team performance, with and without input from the cognitive models has been proposed.

4 References

1. Freed, M. (1998) *Simulating Human Performance in Complex, Dynamic Environments.* Doctoral Dissertation, Northwestern University.

2. John, B. E. (1990) Extensions of GOMS analyses to expert performance requiring perception of dynamic visual and auditory information. In *Proceedings of CHI, 1990* (Seattle, Washington, April 30-May 4, 1990) ACM, New York, 107-115.
3. John, B., Vera, A., Matessa, M., Freed, M., & Remington, R. (2002) Automating CPM-GOMS. *Proceedings of CHI, 2002* (Minneapolis, April 20-25, 2002) ACM, New York.
4. Matessa, M. Vera, A., John, B. E., Remington, R., & Freed, M. (2002) Reusable templates in human performance modeling. *Proceedings of the Twenty-Fourth Annual Conference of the Cognitive Science Society*, August 2002.
5. Pew, R. W, & Mavor, A. S. (1998) Modeling Human and Organizational Behavior: Application to Military Simulations. Panel on Modeling Human Behavior and Command Decision Making: Representations for Military Simulations. Washington, DC: National Academy Press.
6. Remington, R. John, B. E., Vera, A., Matessa, M., Freed, M., Dalal, M., Harris, R., & Dahlman, E. (2002) Apex/CPM-GOMS: Modeling human performance in applied HCI domains. Tutorial materials presented at the *Twenty-Fourth Annual Conference of the Cognitive Science Society*, August 2002

Appendix A:

John, B., Vera, A., Matessa, M., Freed, M., & Remington, R. (2002) Automating CPM-GOMS. *Proceedings of CHI, 2002* (Minneapolis, April 20-25, 2002) ACM, New York.

Automating CPM-GOMS

Bonnie John[1], Alonso Vera[2], Michael Matessa[2], Michael Freed[2], and Roger Remington[2]

[1]Human-Computer Interaction Institute
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA
+1 412 268 7182
bej@cs.cmu.edu

[2]MS 262-4
NASA Ames Research Center
Moffett Field, CA 94035
+1 650 604 6294
avera@arc.nasa.gov

ABSTRACT

CPM-GOMS is a modeling method that combines the task decomposition of a GOMS analysis with a model of human resource usage at the level of cognitive, perceptual, and motor operations. CPM-GOMS models have made accurate predictions about skilled user behavior in routine tasks, but developing such models is tedious and error-prone. We describe a process for automatically generating CPM-GOMS models from a hierarchical task decomposition expressed in a cognitive modeling tool called Apex. Resource scheduling in Apex automates the difficult task of interleaving the cognitive, perceptual, and motor resources underlying common task operators (e.g. mouse move-and-click). Apex's UI automatically generates PERT charts, which allow modelers to visualize a model's complex parallel behavior. Because interleaving and visualization is now automated, it is feasible to construct arbitrarily long sequences of behavior. To demonstrate the process, we present a model of automated teller interactions in Apex and discuss implications for user modeling.

Keywords

GOMS, Apex, Task/User Modeling, Tool Support for Usability Evaluation.

INTRODUCTION AND MOTIVATION

One of the difficulties in developing human interfaces to complex systems is anticipating the response of users to the large space of possible system states and design options. Even extended empirical user testing can fail to uncover serious difficulties. It would be useful to have a computational representation of the user that would allow the designer to simulate user responses to a variety of situations and design options. Though the field is far from having a complete model of the user, several computational modeling approaches have been successful in making accurate predictions of user choices as well as task completion times (e.g., [15, 25, 29, 31, 32]). Of the several architectures available to model human users, the Goals, Operators, Methods, and Selection (GOMS) method [6, 21] has been the most widely used, providing accurate, often zero-parameter, predictions of the routine performance of skilled users in a wide range of procedural tasks [6, 13, 15, 27, 28].

GOMS is meant to model routine behavior. The user is assumed to have methods that apply sequences of operators and to achieve a goal. Selection rules are applied when there is more than one method to achieve a goal. Many routine tasks lend themselves well to such decomposition. Decomposition produces a representation of the task as a set of nested goal states that include an initial state and a final state. The iterative decomposition into goals and nested subgoals can terminate in primitives of any desired granularity, the choice of level of detail dependent on the predictions required.

Although GOMS has proven useful in HCI, tools to support the construction of GOMS models have not yet come into general use. Several tools have emerged from the research world, e.g., QGOMS [3], CATHCI [30], GLEAN [24]. All of these tools aid the modeler to some extent, but all have drawbacks that prevent them from being heavily used in design practice today [2]. In addition, none of them automate any part of the modeling process. However, limited demonstrations of the potential for automating some portions of GOMS modeling have been made [4, 16, 26].

We extend these promising directions with a tool that automates part of the GOMS modeling process using an existing computational architecture, Apex [9, 10]. Our work differs from that of our predecessors because Apex served not only as an implementation platform but provided new insights into GOMS modeling itself. In addition, the previous work focused on the higher-level members of the GOMS modeling family (KLM, CMN-GOMS and NGOMSL; [21]) whereas our use of Apex emphasizes the lowest-level GOMS modeling technique (CPM-GOMS). Employing reusable templates of behavior, our tool allows the modeler to specify procedural knowledge at a task-level and automates the translation of that knowledge into interleaved cognitive, perceptual and motor operators.

CHI 2002, April 20-25, 2002, Minneapolis, Minnesota, USA.

The next section will introduce key aspects of CPM-GOMS and the procedure for constructing such models by hand. Then we will describe Apex, the insights it afforded, and the procedure for constructing a CPM-GOMS model with that tool. Finally, we will present an example use of the tool and a comparison of the resulting CPM-GOMS model to user data.

CONSTRUCTING CPM-GOMS MODELS

John & Kieras [21] described four varieties of GOMS modeling techniques. Three make the assumption that all operators occur in sequence and usually do not contain operators below the activity level (e.g., type-string, move-and-click-mouse). These three are the original formulation by Card, Moran and Newell [5, 6] termed CMN-GOMS, the Keystroke-Level Model (KLM) also formulated by Card Moran and Newell [6], and NGOMSL [23]. The fourth, called CPM-GOMS [17, 18], uses operators at the level of the Model Human Processor (MHP, [6]) and assumes that operators of the **c**ognitive processor, **p**erceptual processor, and the **m**otor processor can work in parallel to each other subject to information-flow constraints. The first three techniques have been supported by research tools for about a decade, where modelers can draw hierarchical goal decomposition in a tree diagram (QGOMS, [3]), program it in a dedicated programming environment (GLEAN, [24]) or even automatically generate most of the model simply by demonstrating a task (CRITIQUE, [16]).

Unlike the first three GOMS methods, CPM-GOMS human performance predictions are constructed from primitives that include estimates of the time for the elementary cognitive, motor, and perceptual operations. These primitives are hypothesized to underlie actions such as typing a key or moving a mouse. Much of the power of CPM-GOMS to predict skilled behavior comes from its ability to model overlapping actions by interleaving cognitive, perceptual, and motor operators. Although it could be argued that CPM-GOMS has been the most economically successful of the GOMS methods (saving a telephone company $2 million per year [15]), it has had no dedicated tool support to date.

Crafting CPM-GOMS Models by Hand

CPM-GOMS models have traditionally been expressed in PERT charts, a representation familiar to project managers. Every operator is represented as a box (a task) with a duration (in milliseconds). If an operator must have information that is the output of another operator, then it is said to be dependent on that operator and must wait for it to complete before starting itself. Likewise, if two operators use the same processor of the MHP (e.g., cognitive processor, vision, or the right hand), one must wait for the previous to complete before starting. Thus, a CPM-GOMS model of a user's task consists of boxes with durations and dependency lines between them. Figure 1 shows a model of a person carefully moving a mouse to a target on the screen and clicking on that target.

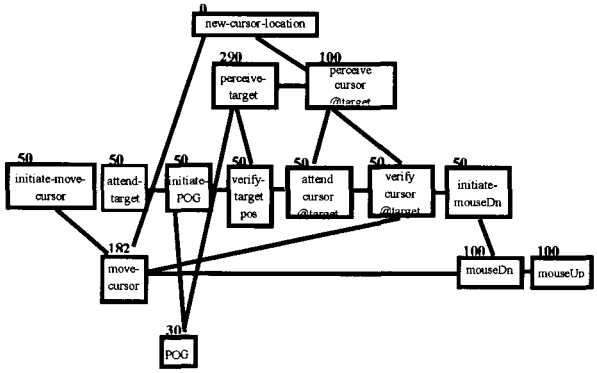

Figure 1: Model of carefully the cursor to a target and clicking the mouse button (adapted from [11]).

Procedure for constructing a CPM-GOMS model with MacProject

Models were created using MacProject, a project management tool originally produced by Apple, improved by Claris, and no longer commercially available. The key feature of MacProject that made it possible for CPM-GOMS models to be constructed is that pre-established patterns of operators could be stored in a library file and then copy-and-pasted into a new canvas, preserving all relevant information about the patterns (e.g., duration, dependencies, position on the page). These patterns, which we called "templates" [20], were of commonly recurring task-level activities in HCI. Each template was very short, some encompassed just a fraction of a second and others were up to several seconds. Templates exist for HCI tasks including typing, visually getting information from a screen (with or without eye-movements), pressing a single key, having a short conversation, etc. The pattern shown in Figure 1 for selecting a target is an example of a template.

To build a CPM-GOMS model, the modeler would start with a hierarchical goal decomposition, usually in the form of a CMN-GOMS model. This goal decomposition would continue until the leaves formed a sequence of keystroke-level activities necessary to complete the task. After completing the goal decomposition, the modeler would choose the templates that achieve the activities and copy and paste them into a blank workspace. The modeler then drew dependency lines between operators from adjoining templates that use the same processor, i.e., from one template's last cognitive operator to the next template's first cognitive operator, etc. Since each template was on the order of a second long, an interesting model would include scores of templates and be comprised of hundreds of MHP-level operators and their dependencies.

After copying the appropriate templates into the model, each operator in each template had to be given a unique name to allow the modeler to keep track of the model as they scrolled through many screens of MacProject. Furthermore, the modeler had to remember to fill in durations for some of the

operator-boxes because the actual duration of the operator varied with the task situation. Many modelers, novice and experienced alike, missed an operator or two in this step, an error that propagates and exacerbates problems throughout the rest of the modeling process.

After all templates are copied in, joined together serially, and customized to the task being modeled, they form a complete PERT chart for the task and MacProject displays the critical path (longest path) for task. At this point, the activities embodied in the templates are modeled as occurring in strict sequence. CPM-GOMS gets it predictive power by breaking the assumption of seriality thereby modeling the ability of highly skilled people to think ahead to the next step while completing the current step, essentially doing several things in parallel. To get this effect in CPM-GOMS models, at every juncture between two templates, the modeler had to consider whether to literally break the dependency line drawn in earlier, put an operator ahead of another operator, and reconnect the dependency lines appropriately. A full set of rules to dictate this step has never been articulated. The first consideration was whether there was sufficient slack time in the critical path to insert an operator belonging to a later template between two operators of the current template. However, deciding when it was appropriate to take advantage of that slack time was more of a craft than an engineering science, involving knowledge of the critical path, the task being modeled, psychology, and intuition. Furthermore, the breaking and reconnecting of scores of dependency lines also usually resulted in some errors of omission, which greatly affected the critical path of the final model.

Although a prose description does not do justice to the procedure, the previous paragraphs attempt to convey that crafting CPM-GOMS with MacProject was difficult, labor-intensive, tedious and error-prone. Add to this the fact that MacProject was not designed for tasks at the millisecond level (the modeler had to work in minutes and do time conversion and it did not have a big enough canvas for long tasks) and the process was also frustrating. Even experienced modelers would take hours to model each minute-long task and then put the model away for a few days and revisit it with "new eyes" to find the errors and inconsistencies. The resulting accuracy of the models, their predictive power, and the eventual clarity of presentation was worth the effort through several projects, but the process was always painful.

Automatically Generating CPM-GOMS Models with Apex

Apex is a computational architecture used to model human behavior in complex dynamic tasks. It incorporates a reactive planner [8] providing capabilities that are a superset of those needed to build GOMS models [11]. These capabilities allowed us to map the concepts of CPM-GOMS to those of Apex and implement CPM-GOMS models directly in Apex.

The Apex architecture

Resources. The Apex architecture includes the concept of resources, which map directly to the MHP's processors, and hence to CPM-GOMS models. Resources operate serially within themselves and are thereby occupied by a single task for the duration of that task. Apex currently has memory, vision, gaze, and right/left hand resources that map to the MHP's cognitive, perceptual, and motor processes. It also has facilities for including more resources as needed by more complex tasks. Apex allocates these resources and others to the tasks it is attempting to execute.

Hierarchical goal decomposition. The hierarchical goal structure of a GOMS model can be expressed in Apex with its Procedure Description Language (PDL). In PDL, a procedure (GOMS method) consists of a number of steps. PDL steps are decomposed hierarchically into procedures of simpler steps until those steps bottom out in primitive actions (GOMS operators) that occupy resources. The decision to perform a particular procedure is mediated by a selection operator (GOMS selection rules). The PDL language is similar to the implementation of NGOMSL in GLEAN [24]. However, PDL is closer in philosophy to CMN-GOMS in that it assigns no time to goal manipulation, only the execution of operators.

Step ordering. In PDL, steps can be assigned a strict serial ordering (like CMN-GOMS or NGOMSL) by explicitly setting the precondition of one step to be the completion of the preceding step. However, the Apex architecture also supports parallelism because if no explicit "waitfor" precondition is assigned, steps can run in parallel (subject to resource constraints). This default assumption of parallel activity is essential to CPM-GOMS models. Apex has a third possibility for ordering steps called *priorities*. In PDL, steps can be assigned a priority. When the step contends for use of a resource, its priority is compared to the priorities of other steps also contending for the same resource, and the task with the highest priority wins the resource. In terms of CPM-GOMS, this allows a sort of soft ordering of templates; task T2 should go after task T1 unless T1 is not using the resource required by T2, in which case T2 can take it.

Time assignment. Primitive actions are assigned durations that can be constants or a function of the environment. For example, the mouseDn action in Figure 1 is assigned an empirically determined value of 100 ms, while the move-cursor action is assigned a time calculated by Fitts's Law. The overall time to run several such actions is calculated by Apex, which takes into account when the actions start and what actions may be running in parallel at any particular time.

Expressing CPM-GOMS Templates in PDL

CPM-GOMS templates can be straightforwardly expressed in PDL. For example, the PERT chart template shown in Figure 1 is expressed in PDL in Figure 2. Each box (operator) in Figure 1 is a step in PDL, labeled "c" for cognitive, "p" for perceptual, and "m" for motor. Dependency lines that go between rows are expressed as explicit "waitfors" in the PDL. For example, the move-cursor motor operator (m1) waits for the initiate-move-cursor cognitive operator (c1). Dependency lines in a row of CPM-GOMS operators are implemented at the next lower level below the code in Figure 2 where the

```
(procedure
(index (slow-move-click ?target))
(step c1 (initiate-move-cursor ?target))
(step m1 (move-cursor ?target)
         (waitfor ?c1))
(step c2 (attend-target ?target))
(step c3 (initiate-eye-movement ?target)
         (waitfor ?c2))
(step m2 (eye-movement ?target)
         (waitfor ?c3))
(step p1 (perceive-target-complex ?target)
         (waitfor ?m2))
(step c4 (verify-target-position ?target)
         (waitfor ?c3 ?p1))
(step c5 (attend-cursor-at-target ?target)
         (waitfor ?c4))
(step w1 (WORLD new-cursor-location ?target)
         (waitfor ?m1))
(step p2 (perceive-cursor-at-target ?target)
         (waitfor ?p1 ?c5 ?w1))
(step c6 (verify-cursor-at-target ?target)
         (waitfor ?c5 ?p2))
(step c7 (initiate-click ?target)
         (waitfor ?c6 ?m1))
(step m3 (mouse-down ?target)
         (waitfor ?m1 ?c7))
(step m4 (mouse-up ?target)
         (waitfor ?m3))
(step t1 (terminate)
         (waitfor ?m4)))
```

Figure 2. PDL code for the CPM-GOMS template shown in Figure 1.

primitive operators are assigned to their resources. That is, both the move-cursor operator and the mouse-down operator are assign to the right-hand resource; since that resource can only do one operator at a time a dependency emerges from Apex's architecture.

Notice in Figure 2 that neither c1 (initiate-move-cursor) nor c2 (attend-target) wait for the completion of any step in this template. This is theoretically appropriate because when selecting a target with a mouse a skilled user can start to point before she starts to look at the target, or start to look before she starts to point. The PDL code enforces no dependency between these two cognitive operators; resource constraints will automatically pick the most appropriate operator at run time.

Articulating CPM-GOMS Template Interleaving Rules

By attempting to express CPM-GOMS templates in PDL and create a complete model of an HCI task, we were able to articulate for the first time reliable rules for appropriately interleaving CPM-GOMS operators. These rules depend on templates like the one in Figure 2, where the operators occupy resources assigned in PDL code below the level of the template and inherit priorities from the goal decomposition code above the template.

The details of how these rules work are beyond the scope of this paper. However, roughly, they allow a momentarily free resource to be seized by an operator from a lower-priority template (i.e., later in the sequence) if no operator from a higher-priority template is ready to request it. If the lower-priority operator can complete before an operator from a higher-priority template requests the resource, that lower-priority operator has successfully interleaved. If it cannot complete before the resource is requested, then it is terminated and reset and it must re-compete for the resource at its next opportunity.

The decision process about how to interleave is completely different for the Apex architecture than for the human modeler using MacProject. The modeler uses knowledge of the entire timecourse of the task encoded in the critical path of the PERT chart, while Apex makes it selections at runtime with no foreknowledge of what other operators are waiting for resources. Despite the differences in decision-making mechanisms, the resulting Apex models interleave operators just as MacProjects models do when created by experts in CPM-GOMS.

Procedure for constructing a CPM-GOMS model with Apex

The first step in creating a CPM-GOMS model with Apex is the same as doing it by hand: create a CMN-GOMS goal decomposition. In Apex, this decomposition is formalized in PDL code instead of just being jotted down on paper or being typed into a word processor. As can be seen in Figure 3, the syntax of PDL code is sufficiently lightweight that this formalization is not a crushing burden.

Both methods depend on previously set-up templates of reusable skills like pointing with a mouse or typing. These reusable templates take the form of PERT charts when using MacProject and PDL code when using Apex. These templates are coded by researchers in cognitive modeling not by system designers modeling a particular interface and task set. Thus, the psychological science is "built in" to the templates by experts in psychology and human modeling so that they can be used easily by non-expert modelers. In addition, in Apex, the psychologists also provide the lower-level code assigning operators to actual and virtual resources, which the modeler never need see.

When the PDL goal decomposition reaches the level of the templates, they simply call the appropriate template as a step in the PDL code. Next, the modeler runs the model using Apex's GUI, Sherpa (Figure 3). By default, Sherpa produces a textual trace of the model, showing the time when each operator starts and completes. However, at the press of a button, Sherpa converts that trace into a PERT chart. The resulting PERT chart contains all the operators, their durations, and their dependencies without any further input from the modeler. No bookkeeping, no deleting or drawing dependency lines, no difficult thinking about interleaving.

In addition, Sherpa has some helpful features tailored to the needs of CPM-GOMS modeling. For instance, the PERT chart can be shrunk horizontally to see patterns within the model or stretched to zoom in on the information in particular operators. It can be toggled to either display in standard PERT chart view where the width of each box is determined only by how much text must fit into it, or to a view where the width of the box is proportional to the time it occupies a